UNIVERSITY OF PLYMOUTH
School of Engineering, Computing & Mathematics

Lecture 5: Front-end Frameworks (Angular & React)

Full-Stack Development

Mark Dixon

School of Engineering, Computing and Mathematics

Last Week

- What did we do last week (write down topics, what you can remember)?
  - Did you learn anything (was anything particularly useful or good)?
  - How did the lab session go (were you able to get something running)?
  - What could be better?

Introduction

Today's topics

1. Angular
2. React.js

Session learning outcomes – **by the end of today's lecture you will be able to:**

- Create simple Angular apps using Angular CLI
- Create single-page apps with the Angular router
- Create simple React.js apps

What is Full Stack?

- A Full Stack Developer has:
  - client-side skills
    (HTML, CSS, JavaScript, jQuery, Angular, React)
  - server-side skills
    (Node.JS, ASP.Net, Python, PHP)
  - database skills
    (SQL Server, Postgres, Oracle, MongoDB)

Full-Stack JavaScript

MongoDB

- NoSQL database

Express.js

- Resource mapping for RESTful web APIs

Angular

- Dynamic data binding for web MVC

Node.js

Angular

A client-side framework

Single-page applications

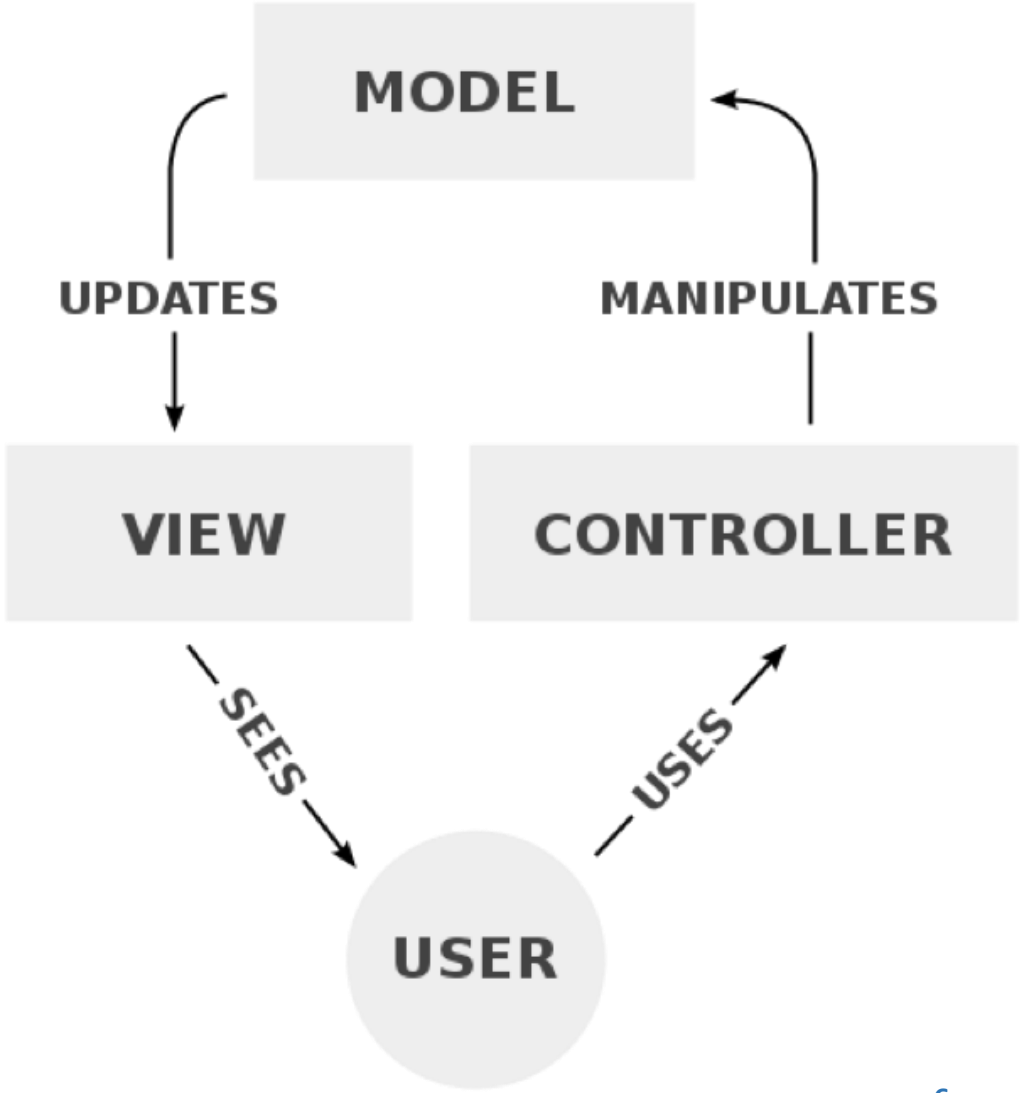
MVC (model-view-controller) capabilities

- Model (scope) – variables
- View (DOM)
- Controller

Extends HTML

- Dynamic two-way data binding

Built on TypeScript (*.ts)



Hello World from Angular

```
ng new my-project-name
```

Use **ng new** command within Angular CLI to create a base project

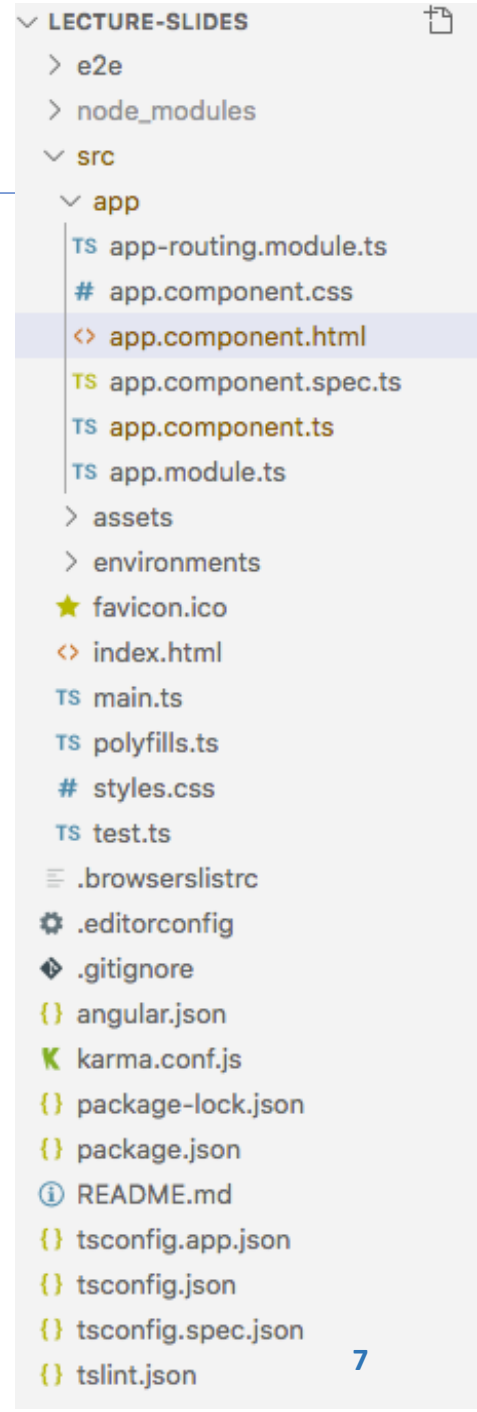
- Much of the code is provided for you
- Adapt the template code to suit your application needs

Hello World app

- Modify **app.component.html** to construct the view

```
<p>{{message}}</p>
```

- Modify **app.component.ts** to set up the model…



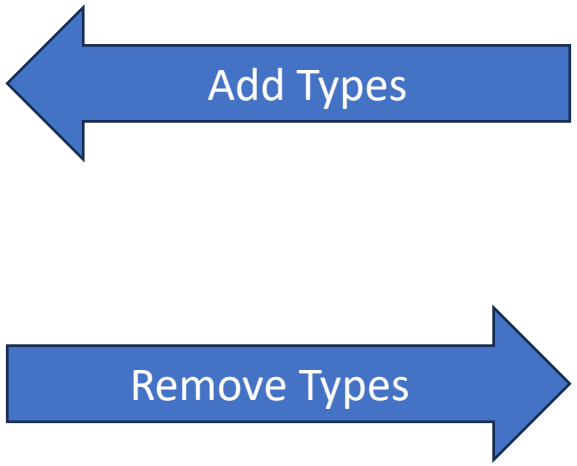
TypeScript

- TypeScript = JavaScript + static typing
- TypeScript – superset of JavaScript
- Therefore, all JavaScript – syntactically correct JavaScript

TypeScript

```
type Result = "pass" | "fail"

function verify(result: Result) {
  if (result === "pass") {
    console.log("Passed")
  } else {
    console.log("Failed")
  }
}
```



JavaScript

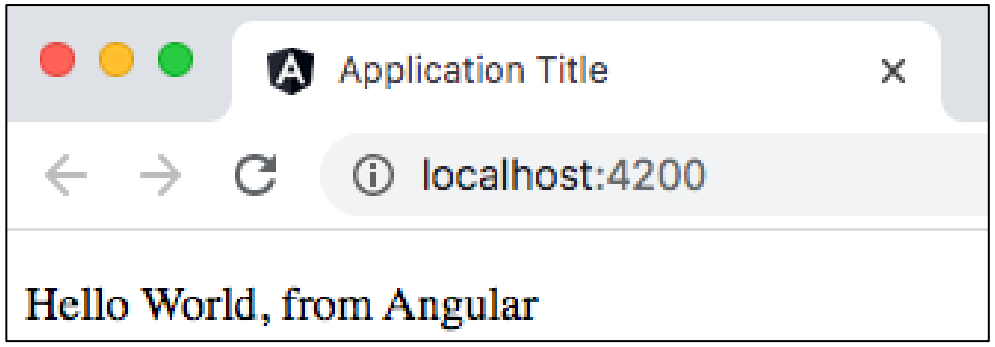
```
function verify(result) {
  if (result === "pass") {
    console.log("Passed")
  } else {
    console.log("Failed")
  }
}
```

Hello World from Angular

```
import { Component } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
    selector: 'app-root',
    templateUrl: './app.component.html'
    styleUrls: ['./app.component.css']
})

export class AppComponent {
    title = 'lecture-slides';
    message = "Hello World, from Angular";
}
```



This is the only change to the generated code

Angular: Project Size

- Angular imports a large number of NPM modules
- Makes project folder large
  (most of this is in node_modules folder)

Size:	276 MB (290,081,978 bytes)
Size on disk:	317 MB (333,238,272 bytes)
Contains:	41,124 Files, 3,691 Folders

Model and view

Define (or obtain) the data in the model

```
export class AppComponent {
    title = 'lecture-slides';

    students = [
        {name: "Abdoulaye Willey", course: "Computer Science"},
        {name: "Lisette Dennis", course: "Computing"},
        {name: "Chen Yating Liao", course: "Computng & Software Development"},
        {name: "Marie Kendall", course: "Computing & Games Development"},
        {name: "Brandon Weeber", course: "Games Development Technologies"},
    ]
}
```

Model and view

Reference from within the view

- Use ***ngFor** to loop over the array of students
- Embed each student within a **table row** element

```
<table>
    <tr><th>Student</th><th>Course</th></tr>
    <tr *ngFor="let student of students">
        <td>{{ student.name }}</td>
        <td>{{ student.course }}</td>
    </tr>
</table>
```

Student	Course
Abdoulaye Willey	Computer Science
Lisette Dennis	Computing
Chen Yating Liao	Computing & SD
Marie Kendall	CGD
Brandon Weeber	GDT

Filtering data

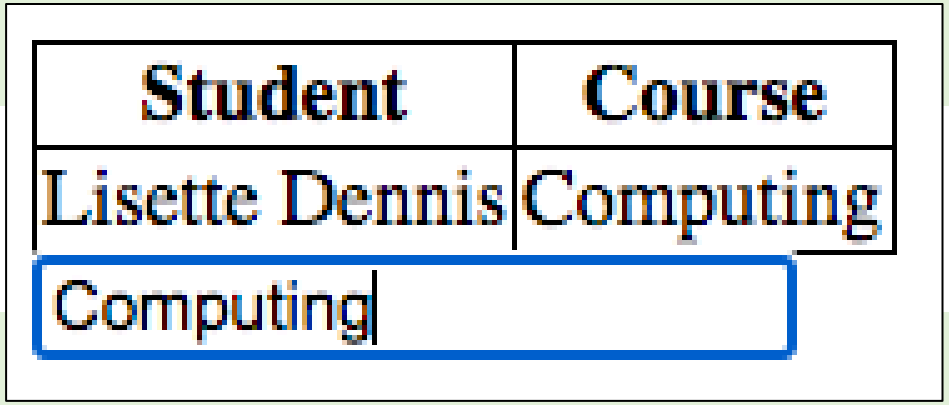
```
<table>
    <tr><th>Student</th><th>Course</th></tr>
    <ng-container *ngFor="let student of students">
        <tr *ngIf="student.course == 'Computer Science'">
            <td>{{ student.name }}</td>
            <td>{{ student.course }}</td>
        </tr>
    </ng-container>
</table>
```

Student	Course
Abdoulaye Willey	Computer Science

Dynamically filtering data

app.component.html

```
<table>
    <tr><th>Student</th><th>Course</th></tr>
    <ng-container *ngFor="let student of students">
        <tr *ngIf="student.course == filter">
            <td>{{ student.name }}</td>
            <td>{{ student.course }}</td>
        </tr>
    </ng-container>
</table>
<input [(ngModel)]="filter" />
```



app.component.ts

```
export class AppComponent {
    filter: string = '';
}
```

Ajax with Angular

1. Modify **app.module.ts** to include the **HttpClientModule**
2. Include **ngOnInit** method in which the Ajax call is made

app.module.ts

- Add **HttpClientModule** to a list of imports in **app.module.ts**

```
import { HttpClientModule } from '@angular/common/http';
```

```
imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule,
    HttpClientModule
],
```

Ajax with Angular

app.component.ts

- Declare an object to store our data and pass to the view
- Dependency injection – inject the **HttpClient** interface
- Declare a **ngOnInit** method that makes the Ajax call and puts the result into the **students** object

```
export class AppComponent {
    students : Object;

    constructor(public httpClient: HttpClient) {}

    ngOnInit() {
        this.httpClient.get("http://localhost:9000/")
            .subscribe((res) => {
                this.students = res["students"];
            });
    }
}
```

Components

- Divide the application into components – **vertically slice** to combine view and controller
- Promotes re-usability
- A component **myComponent** can be included in a template with **<my-component></my-component>**
- Generate using Angular CLI with **ng generate new template my-component**
- For a component called **Module:**

```
<app-module code="COMP3006" title="Full-Stack Development"></app-module>
<app-module code="COMP2002" title="Artificial Intelligence"></app-module>
```

COMP3006: *Full-Stack Development*

COMP2002: *Artificial Intelligence*

Components

```
import { Component, Input, OnInit } from '@angular/core';

@Component({
    selector: 'app-module',
    templateUrl: './module.component.html',
    styleUrls: ['./module.component.css']
})

export class ModuleComponent implements OnInit {

    @Input() code : string = "";
    @Input() title : string = "";

    constructor() {}

    ngOnInit(): void {}

}
```

Routing

- Works in the same way as in **Express**
- Route between **views**

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { CompOneComponent } from './comp-one/comp-one.component';
import { CompTwoComponent } from './comp-two/comp-two.component';

const routes: Routes = [
    {path: "one", component: CompOneComponent},
    {path: "two", component: CompTwoComponent},
];

@NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule]
})

Export class AppRoutingModule {}
```

Routing

```
ng generate component comp-one
ng generate component comp-two
```

comp-one.component.html

```
<p>Component One</p>
```

comp-one.component.css

```
p { color: red; font-size: 24px; }
```

comp-two.component.html

```
<p>Component Two</p>
```

comp-two.component.css

```
p { color: red; font-size: 24px; }
```

Routing

```
<ul>
    <li><a href="/one">Component One</a></li>
    <li><a href="/two">Component Two</a></li>
</ul>

<router-outlet></router-outlet>
```

React.js

- Free
- Open-source
- Front-end JavaScript library

- Single page applications
- Mobile apps
- Server-rendered apps

- Declarative (vs Imperative)
- Hot reloading

React Hello World

- Create React App (CRA) tool
  - Single terminal command

```
npx create-react-app empty-react
```

  - Takes a while (similar to Angular)
  - Generates quite a large project

Size:	256 MB (268,439,659 bytes)
Size on disk:	299 MB (314,478,592 bytes)
Contains:	39,382 Files, 5,038 Folders

  - To run:

```
npm start
```

Hello World Component

- Create HelloWorld.js file (inside src folder):

```
import React from 'react';

const HelloWorld = () => {

    function sayHello() {
        alert('Hello there!!!');
    }

    return (
        <button onClick={sayHello}>Click me!</button>
    )
};

export default HelloWorld;
```

- Add the following to App.js (inside the className div tags):

```
<HelloWorld />
```

Summary

Angular

- Component-based client-side development
- MVC – bind data to views using templates and controllers
- Use Angular CLI to assist with development